Hierarchical Classification of Web Content

Susan Dumais
Microsoft Research
One Microsoft Way
Redmond, WA 99802 USA
sdumais@microsoft.com

Hao Chen
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776 USA
hchen@cs.berkeley.edu

ABSTRACT

This paper explores the use of hierarchical structure for classifying a large, heterogeneous collection of web content. The hierarchical structure is initially used to train different second-level classifiers. In the hierarchical case, a model is learned to distinguish a second-level category from other categories within the same top level. In the flat non-hierarchical case, a model distinguishes a second-level category from all other second-level categories. Scoring rules can further take advantage of the hierarchy by considering only second-level categories that exceed a threshold at the top level.

We use support vector machine (SVM) classifiers, which have been shown to be efficient and effective for classification, but not previously explored in the context of hierarchical classification. We found small advantages in accuracy for hierarchical models over flat models. For the hierarchical approach, we found the same accuracy using a sequential Boolean decision rule and a multiplicative decision rule. Since the sequential approach is much more efficient, requiring only 14%-16% of the comparisons used in the other approaches, we find it to be a good choice for classifying text into large hierarchical structures.

KEYWORDS

Text classification, text categorization, classification, support vector machines, machine learning, hierarchical models, web hierarchies

INTRODUCTION

With the exponential growth of information on the internet and intranets, it is becoming increasingly difficult to find and organize relevant materials. More and more, simple text retrieval systems are being supplemented with structured organizations. Since the 19th century, librarians have used classification systems like Dewey and Library of Congress subject headings to organize vast amounts of information. More recently, web directories such as Yahoo! and LookSmart have been used to classify web pages. Structured directories support browsing and search, but the manual nature of the directory compiling process makes it difficult to keep pace with the ever increasing amount of information. Our work looks at the use of automatic classification methods to supplement human effort in creating structured knowledge hierarchies.

A wide range of statistical and machine learning techniques have been applied to text categorization, including multivariate regression models [8,22], nearest neighbor classifiers [26], probabilistic Bayesian models [13, 16], decision trees [16], neural networks [22, 25], symbolic rule learning [1, 4], and support vector machines [7,12]. These approaches all depend on having some initial labeled training data from which category models are learned. Once category models are trained, new items can be added with little or no additional human effort.

Although many real world classification systems have complex hierarchical structure (e.g., MeSH, U.S. Patents, Yahoo!, LookSmart), few learning methods capitalize on this structure. Most of the approaches mentioned above ignore hierarchical structure and treat each category or class separately, thus in effect "flattening" the class structure. A separate binary classifier is learned to distinguish each class from all other classes. The binary classifiers can be considered independently, so an item may fall into none, one, or more than one category. Or they can be considered as an m-ary problem, where the best matching category is chosen. Such simple approaches work rather well on small problems, but they are likely to be difficult to train when there are a large number of classes and a very large number of features. By utilizing known hierarchical structure, the classification problem can be decomposed into a set of smaller problems corresponding to hierarchical splits in the tree. Roughly speaking, one first learns to distinguish among classes at the top level, then lower level distinctions are learned only within the appropriate top level of the tree. Each of these sub-problems can be solved much more efficiently, and hopefully more accurately as well.

The use of a hierarchical decomposition of a classification problem allows for efficiencies in both learning and representation. Each sub-problem is smaller than the original problem, and it is sometimes possible to use a much smaller set of features for each [13]. The hierarchical structure can also be used to set the negative set for discriminative training and at classification time to combine information from different levels. In addition, there is some

evidence that decomposing the problem can lead to more accurate specialized classifiers. Intuitively, many potentially good features are not useful discriminators in non-hierarchical representations. Imagine a hierarchy with two top-level categories ("Computers" and "Sports"), and three subcategories within each ("Computers/Hardware", "Computers/Software", "Computers/Chat", "Sports/Chat", "Sports/Soccer", Sports/Football"). In a non-hierarchical model, a word like "computer" is not very discriminating since it is associated with items in "Computers/Software", "Computers/Hardware", and "Computers/Chat". In a hierarchical model, the word "computer" would be very discriminating at the first level. At the second level more specialized words could be used as features within the top-level "Computer" category. And, the same features could be used at the second level for two different top-level classed (e.g., "chat" might be a useful feature for both the category "Sports/Chat", and "Computers/Chat"). Informal failure analyses of classification errors for non-hierarchical models support this intuition. Many of the classification errors are for related categories (e.g., a page about "Sports/Soccer" might be confused with "Sports/Football", thus category specific features should improve accuracy).

Recently several researchers have investigated the use of hierarchies for text classification, with promising results. Our work differs from earlier work in a couple of important respects. First, we test the approach on a large collection of very heterogeneous web content, which we believe is increasingly characteristic of information organization problems. Second, we use a learning model, support vector machine (SVM), that has not previously been explored in the context of hierarchical classification. SVMs have been found to be more accurate for text classification than popular approaches like naïve Bayes, neural nets, and Rocchio [7, 12, 27]. We use a reduced-dimension binary-feature version of the SVM model that is very efficient for both initial learning and real-time classification, thus making it applicable to large dynamic collections. We will briefly review the earlier work and contrast it with ours.

RELATED WORK

Reuters

Much of the previous work on hierarchical methods for text classification uses the Reuters-22173 or Reuters-21578 articles. This is a rather small and tidy collection, and this alone is problematic for understanding how the approaches generalize to larger more complex internet applications. In addition, the Reuters articles are organized into 135 topical categories with no hierarchical structure. To study hierarchical models, researchers have added one level of hierarchical structure manually. Kohler and Sahami [13] generated a small hierarchical subset of Reuters-22173 by identifying labels that tended to subsume other labels (e.g., corn and wheat are subsumed by grain). The largest of their hierarchies consisted of 939 documents organized into 3 top-level and 6 second-level categories. They compared naïve Bayes, and two limited dependency Bayes net classifiers on flat and hierarchical models. Test documents were classified into the hierarchy by first filtering them through the single best matching first level class and then sending them to the appropriate second level. Note that errors made at the first level are not recoverable, so the system has to make k correct classification for a k-level hierarchy. They found advantages for the hierarchical models when a very small number of features (10) were used per class. For larger numbers of features (which will be required in many complex domains) no advantages for the hierarchical models were found.

Weigend et al. [25] also used the Reuters-22173 collection. An exploratory cluster analysis was first used to suggest an implicit hierarchical structure, and this was then verified by human assignments. They created 4 top-level categories (agriculture, energy, foreign exchange, and metals) and a 5th miscellaneous category that was not used for evaluation. The final evaluations were on 37 categories with at least 16 positive training examples. They used a probabilistic approach that frames the learning problem as one of function approximation for the posterior probability of the topic vector given the input vector. They used a neural net architecture and explored several input representations. Information from each level of the hierarchy is combined in a multiplicative fashion, so no hard decisions have to be made except at the leaf nodes. They found a 5% advantage in average precision for the hierarchical representation when using words, but not when using latent semantic indexing (LSI) features.

D'Alessio et al. [6] used the Reuters-21578 articles. The hierarchy they use comes from Hayes and Weinstein's original Reuters experiments [9]. It consists of 5 meta-category codes (economic indicator, currency, corporate, commodity, energy) that include all but three of the original categories. For their experiments they only considered articles that had a single category tag, and the 37 categories with more than 20 positive training examples. Their model requires hard assignment at each branch of the tree. The hierarchical model showed 2-4% improvements in precision and recall over the flat model, and modifications to the hierarchy led to advantages of 2-9%.

Ng et al. [19] also used a hierarchical version of Reuters that consisted of countries at the top level and two topical levels below that, but they did not compare their hierarchical model against a flat model. While all of this work is encouraging, the Reuters collection is small and very well organized compared with many realistic applications.

MeSH, U.S. patents

Some researchers have investigated text classification in domains that have rich hierarchical taxonomies (e.g., MeSH, IDC codes of diseases, U.S. patent codes). The size and complexity of the medical and patent hierarchies are like those used for web content. These hierarchies were

designed for controlled vocabulary tagging and they have been extensively refined over the years. Ruiz and Srinivasan [21] used a hierarchical mixture of experts to classify abstracts within the MeSH sub-category Heart. They learned classifiers at different levels of granularity, with the top-level distinctions serving as "gates" to the lower level "experts". They found small advantages for the hierarchical approach compared with a flat approach. Larkey [15] compared hierarchical and flat approaches for classifying patents in the Speech Signal Processing sub-category. She found no multilevel algorithms that performed significantly better than a flat one which chooses among all the speech classes.

Web

McCallum, et al. [17] describe some interesting experiments on three hierarchical collections -- Usenet news, the Science sub-category of Yahoo!, and company web pages. The Yahoo! sub-collection is most closely related to our experiments, although it is less diverse since all items come from the same top-level category. They used a probabilistic Bayesian framework (naïve Bayes), and a technique called "shrinkage" to improve parameter estimates for the probability of words given classes. The idea is to smooth the parameter estimate of a node by interpolation with all its parent nodes (given by the hierarchical organization of classes). This is especially useful for classes with small numbers of training examples. For the Usenet collection, the best performance and largest advantages of the hierarchical model over the flat model are observed for large numbers of features (10,000). For the Yahoo! sub-collection, accuracy is low and there is a much smaller difference between two approaches

Mladenic and Grobelnik [18] examined issues of feature selection for hierarchical classification of web content, but they did not compare hierarchical models with flat non-hierarchical models. Chakrabarti et al. [2] also experimented with web content as well as patent and Reuters data. They use hierarchical structure both to select features and to combine class information from multiple levels of the hierarchy. For the non-hierarchical case, they use a simple vector method with term weighting and no feature selection. They find no advantages for the hierarchical approach in Reuters, a small advantage in the patents collection, and did not test the difference for the Yahoo! sub-collection. But, because they used different representations, it is difficult to know whether differences are due to the number of features, feature selection, or the hierarchical classification algorithm.

Our work explores the use of hierarchies for classifying very heterogeneous web content. We use SVMs, which have been found to be efficient and effective for text classification, but not previously explored in the context of hierarchical classification. The efficiency of SVMs for both initial learning and real-time classification make them applicable to large dynamic collections like web content. We use the hierarchical structure for two purposes. First, we train second-level category models using different contrast sets (either within the same top-level category in the hierarchical case, or across all categories in the flat non-hierarchical case). Second, we combine scores from the top- and second-level models using different combination rules, some requiring a threshold to be exceeded at the top level before second level comparisons are made. The sequential approach is especially efficient for large problems, but how it compares in accuracy is not known. All of our models allow for each test item to be assigned to multiple categories.

WEB DATA SET AND APPLICATION

Application – Classifying web search results

We are interested in moving beyond today's ranked lists of results by organizing web search results into hierarchical categories. HCI researchers have shown usability advantages of structuring content hierarchically [3, 10, 14]. For information as varied and voluminous as the web it is necessary to create these structures automatically. Several groups have explored methods for *clustering* search results. Zamir and Etzioni [29] grouped web search results using suffix tree clustering, and Hearst and Pedersen [11] used Scatter/Gather to organize and browse search results. While these methods show some promise, the computational complexity of clustering algorithms limits the number of documents that can be processed, and generating names for the resulting categories is a challenge.

The basic idea behind our work is to use *classification* techniques to automatically organize search results into existing hierarchical structures. Classification models are learned offline using a training set of human-labeled documents and web categories. Classification offers two advantages compared to clustering – run time classification is very efficient, and manually generated category names are easily understood. To support our goal of automatically classifying web search results two constraints are important in understanding the classification problems we studied. First, we used just the short summaries returned from web search engines since it takes too long to retrieve the full text of pages in a networked environment. These automatically generated summaries are much shorter than the texts used in most classification experiments, and they are much noisier than other document surrogates like abstracts that some have worked with. Second, we focused on the top levels of the hierarchy since we believe that many search results can be usefully disambiguated at this level. We also developed an interface that tightly couples search results and category structure, and found large preference and performance advantages for automatically classified search results (see Chen and Dumais [3] for details). For the experiments reported in this paper, we focused on the top two levels of the hierarchy, and used short automatically generated summaries of web pages.

LookSmart Web Directory

For our experiments, we used a large heterogeneous collection of pages from LookSmart's web directory [30]. At the time we conducted our experiments in May 1999, the collection consisted of 370597 unique pages that had been manually classified into a hierarchy of categories by trained professional web editors. There were a total of 17173 categories organized into a 7-level hierarchy. We focused on the 13 top-level and 150 second-level categories.

Training/Test Selection

We selected a random 16% sample of the pages for our experiments. We picked this sampling proportion to ensure that even the smallest second-level categories would have some examples in our training set. We used 50078 pages for training, and 10024 for testing. Table 1 shows the number of pages in each top level category. Top level categories had between 578 and 11163 training examples, and second-level categories had between 3 and 3141 training examples. The wide range in number of pages in each category reflects the distribution in the LookSmart directory. Pages could be classified into more than one category. On average, pages were classified into 1.20 second-level categories and 1.07 first-level categories.

Category Name	Total	Train	Test
Automotive	4982	578	114
Business & Finance	31599	3508	703
Computers & Internet	46000	5718	1126
Entertainment & Media	88697	11163	2159
Health & Fitness	25380	3500	722
Hobbies & Interests	22959	3227	682
Home & Family	11484	1373	280
People & Chat	35157	3309	682
Reference & Education	58002	5574	1175
Shopping & Services	19667	2122	423
Society & Politics	38968	4855	946
Sports & Recreation	23559	3081	640
Travel & Vacations	43685	5409	1091
Total Unique	370597	50078	10024

Table 1- Training and Test Samples by Category

Pre-processing

A pre-processing module extracted plain text from each web page. In addition, the title, description and keywords fields from the META tag, and the ALT field from the IMG tag were also extracted if they existed because they provide useful descriptions of the web page. If a web page contained frames, text from each frame was extracted.

Since we were interested in classifying web search results, we needed to work with just short summary descriptions of web pages. We automatically generated summaries of each page and used this for evaluating our classification algorithms. Our summaries consisted of the title, the keywords, and either the description tag if it existed or the first 40 words of the body otherwise.

We did minimal additional processing on the text. We translated upper to lower case, used white space and punctuation to separate words, and used a small stop list to omit the most common words. No morphological or syntactic analyses were used.

After preprocessing, a binary vector was created for each page indicating whether each term appeared in the page summary or not. For each category 1000 terms were selected based on the mutual information between the term and category (details below). Unlike many text retrieval applications, term frequency and document length are not taken into account in this representation, because earlier experiments showed good performance with the binary representation for SVMs [7], and this representation improves efficiency.

TEXT CLASSIFICATION USING SVMs

Text classification involves a training phase and a testing phase. During the training phase, a large set of web pages with known category labels are used to train a classifier. An initial model is built using a subset of the labeled data, and a holdout set is used to identify optimal model parameters. During the testing or operational phase, the learned classifier is used to classify new web pages.

A support vector machine (SVM) algorithm was used as the classifier, because it has been shown in previous work to be both very fast and effective for text classification problems [7, 12, 27]. Vapnik introduced SVMs in his work on structural risk minimization [23, 24], and there has been a recent surge of interest in SVM classifiers in the learning community. In its simplest form, a linear SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin. The margin is the distance from the hyperplane to the nearest of the positive and negative examples.

Figure 1 shows an example of a simple two-dimensional problem that is linearly separable.

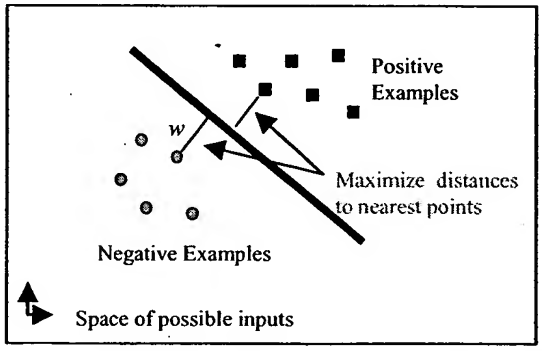

Figure 1 - Graphical Representation of a Linear SVM

In the linearly separable case maximizing the margin can be expressed as an optimization problem:

minimize $\frac{1}{2}\|\vec{w}\|^2$ subject to $y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1, \forall i$

where x_i is the ith training example and y_i is the correct output of the SVM for the ith training example.

In cases where points are not linearly separable, slack variables are introduced that permit, but penalize, points that fall on the wrong side of the decision boundary. For problems that are not linearly separable, kernel methods can be used to transform the input space so that some non-linear problems can be learned. We used the simplest linear form of the SVM because it provided good classification accuracy, and is fast to learn and apply.

Platt [20] developed a very efficient method for learning the SVM weights. His sequential minimal optimization (SMO) algorithm breaks the large quadratic programming (QP) problem down into a series of small QP problems that can be solved analytically. Additional efficiencies can be realized because the training sets used for text classification are sparse and binary. Thus the SMO algorithm is nicely applicable for large feature and training sets.

Once the weights are learned, new items are classified by computing $\vec{w} \cdot \vec{x}$, where $\vec{w}$ is the vector of learned weights, and $\vec{x}$ is the input vector for a new document. With the binary representation we use, this amounts to taking the sum of the weights for features present in a document and this is very efficient.

After training the SVM, we fit a sigmoid to the output of the SVM using regularized maximum likelihood fitting, so that the SVM produces posterior probabilities that are directly comparable across categories.

Feature Selection

For reasons of both efficiency and efficacy, feature selection is often used when applying machine learning methods to text categorization. We reduce the feature space by eliminating words that appear in only a single document (hapax legomena), then selecting the 1000 words with highest mutual information with each category. Yang and Pedersen [28] compared a number of methods for feature selection. We used a mutual information measure (Cover and Thomas [5]) that is similar to their information gain measure. The mutual information MI(F, C) between a feature, F, and a category, C, is defined as:

$$MI(F,C) = \sum_{F \in \{f,\bar{f}\}} \sum_{C \in \{c,\bar{c}\}} P(F,C) \log \frac{P(F,C)}{P(F)P(C)}$$

We compute the mutual information between every pair of features and categories.

For the non-hierarchical case, we select the 1000 features with the largest MI for each of the 150 categories (vs. all the other categories). For the hierarchical case, we select 1000 features with the largest MI for each of the 13 top-level categories, and also select the 1000 features for each of the 150 second-level categories (vs. the categories that share the same top-level category). We did not rigorously explore the optimum number of features for this problem, but these numbers provided good results on a training validation set so they were used for testing. In all cases, the selected features are used as input to the SVM algorithm that learns the optimum weights for the features, $\vec{w}$.

SVM Parameters

In addition to varying the number of features, SVM performance is governed by two parameters, C (the penalty imposed on training examples that fall on the wrong side of the decision boundary), and p (the decision threshold). The default C parameter value (0.01) was used.

The decision threshold, p, can be set to control precision and recall for different tasks. Increasing p, results in fewer test items meeting the criterion, and this usually increases precision but decreases recall. Conversely, decreasing p typically decreases precision but increases recall. We chose p so as to optimize performance on the F_1 measure on a training validation set. For the flat non-hierarchical models, p=0.5. For the hierarchical models, p=0.2 for the multiplicative decision rule, and p_1=0.2 (first level) and p_2=0.5 (second level) for the Boolean decision rule.

RESULTS

As just described decision thresholds were established on a training validation set. For each category, if a test item exceeds the decision threshold, it is judged to be in the category. A test item can be in zero, one, or more than one categories. From this we compute precision (P) and recall (R). These are micro-averaged to weight the contribution of each category by the number of test examples in it. We used the F measure to summarize the effects of both precision and recall. With equal weights assigned to precision and recall, $F_1 = 2*P*R/(P+R)$.

For each test example, we compute the probability of it being in each of the 13 top-level categories and each of the 150 second-level categories. Recall that for the non-hierarchical case, models were learned (and features chosen) to distinguish each category from all other categories, and that for the hierarchical case, models were learned (and features chosen) to distinguish each category from only those categories within the same top-level category. We explored two general ways to combine probabilities from the first and second level for the hierarchical approach. In one case we first set a threshold at the top level and only match second-level categories that pass this test – that is we compute a Boolean function $P(L1)\&\&P(L2)$. In order to be correctly classified, a test instance must satisfy both constraints. This method is quite efficient, since large numbers of second level categories do not need to be tested. In the other case, we compute $P(L1)*P(L2)$. This approach allows matches even though the scores at one level fall below threshold. Although the non-hierarchical models are not trained to use top-level information, we can compute the same probabilities for this case as well.

Learned Features

There are many differences in the learned features for the hierarchical and non-hierarchical models. For example, the feature "sports" is useful in distinguishing the top-level category Sports & Recreation from the other top-level categories. The feature "sports" is also useful for distinguishing between some second-level categories -- it has a high weight for News & Magazines (under Society & Politics), SportingGoods (under Shopping & Services), and Collecting (under Hobbies & Interests). But, the feature "sports" is not selected for any categories using the flat non-hierarchical approach. It is difficult to study this systematically, but the different learning approaches are serving an important role in feature selection and weighting.

F_1 Accuracy

Top Level

The overall F_1 value for the 13 top-level categories is .572. Classifying short summaries of very heterogeneous web pages is a difficult task, so we did not expect to have exceptionally high accuracy. Performance on the original training set is only .649, so this is a difficult learning task and generalization to the test set is quite reasonable. This level of accuracy is sufficient to support some web search tasks [3], and could also be used as an aid to human indexers. In addition, we know that we can improve the absolute level of performance by 15%-20% using the full text of pages, and by optimizing the *C* parameter. What is of more interest to us here is the comparative performance of hierarchical and non-hierarchical approaches.

Second Level, Non-Hierarchical (Baseline)

The overall F_1 value for the 150 second-level categories, treated as a flat non-hierarchical problem, is .476. There is a drop in performance in going from 13 to 150 categories. Performance for the 150 categories is better than the .364 value reported by McCallum et al. [17], but it is difficult to compare precisely because they use average precision not F_1, the categories themselves are different, and they use the full text of pages. The .476 non-hierarchal value will serve as the baseline for looking at the use of hierarchical models.

Performance varies widely across categories. The five categories with highest and lowest F_1 scores (based on at least 30 test instances) are shown in Table 2.

The most difficult categories to learn models for are those based, at least in part, on non-content distinctions. It might be useful to learn models for genre (e.g., for kids, magazines, web) in addition to our current use of content-based models.

Second Level, Hierarchical

As described above, we examined both a multiplicative scoring function, *P(L1)*P(L2)*, and a Boolean scoring function, *P(L1)&&P(L2)*. The former allows for matches that fall below individual thresholds, and the latter requires that the threshold be exceeded at every level. Both scoring rules allow items to be classified into multiple classes.

The overall F_1 value for the *P(L1)*P(L2)* scoring function is .495, at the threshold of p=0.20 established on the validation set. This represents a 4% improvement over the baseline flat models, and is statistically significant using a sign test, p<.01. The overall F_1 value for the *P(L1)&&P(L2)* scoring function is .497, at the thresholds of p_1=0.20 and p_2=0.50 established on the validation set. This again is a small improvement over the non-hierarchical model, and is significantly different than the baseline using a sign test, p<0.01. Somewhat surprisingly, there is no difference between the sequential Boolean rule and the multiplicative scoring rule. The added efficiencies that can be obtained with the hierarchical model and the Boolean rule make it a good overall choice, as we discuss in more detail below.

An upper bound on the performance of these hierarchical models can be obtained by assuming that the top-level classification is correct (i.e., *P(L1)=1*). In this case, performance is quite good with an F_1 value of .711, so there is considerable room for improvement.

	Category Name
Best F1	
0.841	*Health & Fitness/Drugs & Medicines*
0.797	*Home & Family/Real Estate*
0.781	*Reference & Education/K-12 Education*
0.750	*Sports & Recreation/Fishing*
0.741	*Reference & Education/Higher & Cont. Ed.*
Worst F1	
0.034	*Society & Politics/World Cultures*
0.088	*Home & Family/For Kids*
0.122	*Computers & Internet/News & Magazines*
0.131	*Computers & Internet/Internet & the Web*
0.133	*Business & Finance/Business Professions*

Table 2 - Best and worst F1 score by category

Efficiency

Offline training

As noted earlier, linear SVM models can be learned quite efficiently with Platt's SMO algorithm. The total training time for all the models described in the paper was only about 30 minutes. Training time for all the non-hierarchical models on 50078 examples in each of 150 categories was 729 CPU seconds. Training time for all 150 hierarchical second-level models was 128 CPU seconds. Training is faster here because the negative set is smaller, including only items in the same top-level category. Training time for the 13 top-level categories was 1258 CPU seconds. All times were computed on a standard 266MHz Pentium II PC running Windows NT.

Online evaluation

At run time we need to take the dot product of the learned weight vector for each category and the feature vector for the test item. The cost of this operation depends on the number of features and the number of categories.

Efficiency can be improved if some category comparisons can be omitted without hurting accuracy. As we have seen above, the Boolean decision function accomplishes this with no decrease in classification accuracy. For the non-hierarchical model, each test instance must be compared with all 150 second-level categories. For the hierarchical model with the multiplicative scoring rule, each test instance must be compared to all 13 first-level categories and all 150 second-level categories. For the Boolean scoring rule, each test instance must be compared to all 13 first-level categories but only to second-level categories that pass the first-level criterion. For the top-level threshold value of p=0.20, only 7.4% of the second-level categories need to be examined. Top-level comparisons are still required for the Boolean decision rule, so overall 14.8% of the comparisons used for the multiplicative rule and 16.1% of the comparisons used for the non-hierarchical model are required, both representing a considerable savings at evaluation time.

CONCLUSION

The research described in this paper explores the use of hierarchical structure for classifying a large, heterogeneous collection of web content to support classification of search results. We used SVMs, which have been found to be an efficient and effective learning method for text classification, but not previously explored for hierarchical problems. We use the hierarchical structure for two purposes. First, we train second-level category models using different contrast sets (either within the same top-level category in the hierarchical case, or across all categories in the non-hierarchical case). Second, we combine scores from the top- and second-level models using different combination rules, some requiring a threshold to be exceeded at the top level before comparisons at the second level are made.

We found small advantages in the F_1 accuracy score for the hierarchical models, compared with a baseline flat non-hierarchical model. We found no difference between a multiplicative scoring function and a sequential Boolean function for combining scores from the two levels of the hierarchy. Since the sequential Boolean approach is much more efficient, requiring only 14%-16% of the number of comparisons, we find it to be a good choice.

There are a number of interesting directions for future research. We should be able to obtain further advantages in efficiency in the hierarchical approach by reducing the number of features needed to discriminate between categories within the same top-level category. Koller and Sahami [13] have shown that one can dramatically reduce the number of features for simple Reuters categories without decreasing accuracy too much, and this is definitely worth trying with our content and learning algorithm. There are some preliminary indications that this will work in our application. The SVM model assigns a weight of 0 to features that are not predictive (and could thus be omitted). We find a larger number of features with 0 weights in the hierarchical case (137 per 1000) than the non-hierarchical case (85 per 1000). There are also many near-zero weights that contribute little to the overall score. Varying the number of features for further efficiency gains seems promising to try with our content and learning model.

The sequential decision model provides large efficiency gains, so it will be important to see how it generalizes to more than two levels, where error cascading may be more of an issue. There are at least two ways to address the problem. One involves improved classifiers and methods for setting appropriate decisions thresholds at multiple levels. Another approach is through the use of interactive interfaces so that people are involved in making some of the critical decisions. This is a rich space for further exploration.

Our research adds to a growing body of work exploring how hierarchical structures can be used to improve the efficiency and efficacy of text classification. We worked with a large, heterogeneous collection of web content -- a scenario that is increasingly characteristic of the information management tasks that individuals and organizations face. And, we successfully extended SVM models to an application that takes advantage of hierarchical structure, for both category learning and run time efficiencies.

ACKNOWLEDGMENTS

We are grateful to John Platt for help with the Support Vector Machine code, and to four anonymous reviewers for their comments.

REFERENCES

1. Apte, C., Damerau, F. and Weiss, S. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems, 12(3),* 233-251, 1994.
2. Chakrabarti, S., Dom, B., Agrawal, R. and Raghavan, P.. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal 7,* 163-178, 1998.
3. Chen, H. and Dumais, S. Bringing order to the web: Automatically categorizing search results. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'00),* 145-152, 2000.
4. Cohen, W.W. and Singer, Y. Context-sensitive learning methods for text categorization *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96),* 307-315, 1996.
5. Cover, T. and Thomas, J. *Elements of Information Theory.* Wiley, 1991.
6. D'Alessio, S., Murray, M., Schiaffino, R. and Kershenbaum, A. Category levels in hierarchical text

categorization. *Proceedings of EMNLP-3, 3rd Conference on Empirical Methods in Natural Language Processing*, 1998.

7. Dumais, S. T., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM'98)*, 148-155, 1998.
8. Fuhr, N., Hartmanna, S., Lustig, G., Schwantner, M., and Tzeras, K. Air/X – A rule-based multi-stage indexing system for lage subject fields. *Proceedings of RIAO'91*, 606-623, 1991.
9. Hayes, P.J. and Weinstein, S.P. CONSTRUE: A System for Content-Based Indexing of a Database of News Stories. *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.
10. Hearst, M., and Karadi, C. Searching and browsing text collections with large category hierarchies. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'97), Conference Companion*, 1997.
11. Hearst, M. and Pedersen, J. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. *Proceedings of 19th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, 1996.
12. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. *Proceedings of European Conference on Machine Learning (ECML'98)*, 1998
13. Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 170-178, 1997.
14. Landauer, T., Egan, D., Remde, J., Lesk, M., Lochbaum, C., and Ketchum, D. Enhancing the usability of text through computer delivery and formative evaluation: The SuperBook project. *Hypertext – A Psychological Perspective*. Ellis Horwood, 1993.
15. Larkey, L. Some issues in the automatic classification of U.S. patents. In *Working Notes for the AAAI-98 Workshop on Learning for Text Categorization*, 1998.
16. Lewis, D.D. and Ringuette, M.. A comparison of two learning algorithms for text categorization. *Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 81-93, 1994.
17. McCallum, A., Rosenfeld, R., Mitchell, T. and Ng, A. Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of the Fifteenth International Conference on Machine Learning, (ICML-98)*, 359-367, 1998.
18. Mladenic, D. and Grobelnik, M. Feature selection for classification based on text hierarchy. *Proceedings of the Workshop on Learning from Text and the Web*, 1998.
19. Ng, H.T., Goh, W.B. and Low, K.L, *Proceedings of 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, 67-73, 1997.
20. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods –Support Vector Learning*. B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1999.
21. Ruiz, M.E. and Srinivasan, P. Hierarchical neural networks for text categorization. *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 281-282, 1999.
22. Schütze, H., Hull, D. and Pedersen, J.O. A comparison of classifiers and document representations for the routing problem. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, 229-237, 1995.
23. Vapnik, V., *Estimation of Dependencies Based on Data [in Russian]*, Nauka, Moscow, 1979. (English translation: Springer Verlag, 1982.)
24. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.
25. Weigend, A.S., Wiener, E.D. and Pedersen, J.O. Exploiting hierarchy in text categorization. *Information Retrieval, 1(3)*, 193-216, 1999.
26. Yang, Y. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, 13-22, 1994.
27. Yang, Y. and Lui, Y. A re-examination of text categorization methods. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 42-49, 1999.
28. Yang, Y. and Pedersen, J.O. A comparative study on feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 412-420, 1997.
29. Zamir, O. and Etzioni, O. Web document clustering: A feasibility demonstration. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 46-54, 1998.
30. http://www.looksmart.com